Research Task / Overview

• Understanding software quality evolution, and conflicts and synergies among software quality attributes by analyzing impactful changes.
• Utilizing multiple programming analysis techniques to conduct multi-perspective analysis of software quality.
• Capitalizing on cloud services to analyze revision histories of large families of open-source software systems.
• Conducting large-scale replicable empirical studies.
• Providing interactive desktop and web interfaces to illustrate software quality evolution.

Methodology

How to identify impactful changes?

• Version Control System (i.e., Git)
  - Tracks changes and contains fine-grained information, such as when change occurs, how, and by who.
  - Commit impact analysis
  - Analyzing software quality before and after commits that change the source code of the main module of software.

How to evaluate change in quality?

• Static Analysis Techniques
  - Analyze software without running it. Report Parities
  - Retrieve quality metrics from reports and store them in a uniform relational schema.

How to explore the data?

• Interactive Desktop and Web Interfaces
  - Evolution trend of a metric.
  - Impact of each developer.
  - Evolution graph of a metric.
  - Co-evolution of multiple metrics.

How to scale?

• Automated Cloud-Based Infrastructure
  1. Retrieve a subject system’s meta-data (e.g., number of contributors) as well as its commit history from Github.
  2. Distribute hundreds of revisions (i.e. official releases and/or revisions created by commits) on multiple cloud instances.
  3. Run static/dynamic programming analysis techniques on each revision.
  4. Collect and parse the artifacts generated by programming analysis techniques to extract quality attributes.
  5. Run various statistical analysis on software quality evolution. The entire analysis workflow is automated.

Goals & Objectives

• Helping organizations determine which divisions and project types have better or worse quality; which quality attributes are being achieved poorly or well; and how these trends correlate with customer satisfaction and total cost of ownership.
• Helping project managers better understand which types of projects or personnel contribute most to quality problems or excellence, and which types of project evolution correlate with which types of quality increase or decrease.
• Helping developers continuously monitor software quality and improve software maintainability.
• Helping acquisition programs evaluate system quality.

Data & Analysis

Research Questions

• RQ1: To what extent do developers commit impactful changes?
• RQ2: To what extent and how do impactful commits break the code quality?
• RQ3: To what extent do impactful commits affect software quality attributes?
• RQ4: Should developers rely on a single quality metric as a change indicator?

Data Collection

- Collected the metadata of all Apache projects via Github API
- Name, # of commits, programming language, and last update date
- Subject system selection criteria:
  - Java, updated in 2017, the main module exists, no nontrivial prerequisite for compilation, at least 100 compilable different revisions, and less than 3K commits.
- Scale of the analysis:
  - 3 techniques and analysis techniques, 9 metrics, 38 systems, 19580 impactful commits and revisions, 643 impactful developers, 586 MSLOC, 15 years timespan.

Results’ Highlight

• On average, 48% of all commits impact the main module, and 69% of all developers contribute to the source code in that main module.
• On average, 2% of the impactful commits are not compilable.
• Different quality attributes may change even if the code count does not change.
• Only 5.7% impactful commits change V1, 1.9% change SG, and 2.4% change FG.
• Although security metrics change less frequently, it is crucial to utilize them as they can reveal the introduction of different kinds of security problems.

Tips to Avoid Introducing Complex Error

• When Contributing to Open Source Projects
  - Avoid using snapshot of dependencies!
  - Compile in a new environment:
    - Contributing alone.
    - Changing build files.
    - Adding new files.
    - Conducting large refactoring.
    - Don’t commit too early and too often!

Translating Research into Practice

- We recently delivered advanced tool assessments tutorials to front line acquisition engineers of a major governmental entity.
- This led to an in-depth analysis of the quality aspects of an open source software complex for decisions regarding quality, safety, and security "sprints" and "taints" to assess an acquisition program of an unmanned system.

Follow-up Study

- “How Do Contributors’ Involvement Influence Open Source Systems” [2]. In this paper, we investigated whether variations in contributors level of involvement shows variation in their contribution type and its quality with an emphasis on technical debt.

References


Contacts/References

References:


Contact Info:

- Ski Bloom Walk, S&L 327, Los Angeles, CA 90089
- +1 213 400 7256
- pbehnamgh@usc.edu

Towards Better Understanding of Conflicts and Synergies Among Software Quality Attributes By Analysis of Abundant Data

Pouyeh Behnamghader
USC Center for Systems and Software Engineering

Future Research

- Difference between developers in terms of impact on software quality.
- Studies with dynamic analysis techniques and regression tests.
- Effect and intent of the changes on software quality.
- Software-quality defect prediction models.